TEC | Tecnológico de Costa Rica

Programa del curso IC-6821

Diseño de Software

Escuela de Computación
Carrera de Ingeniería en Computación, Plan 410

I parte: Aspectos relativos al plan de estudios

1 Datos generales

Nombre del curso:	Diseño de Software
Código:	IC-6821
Tipo de curso:	Teórico – práctico
Electivo o no:	No
Nº de créditos:	4
Nº horas de clase por semana:	4
Nº horas extraclase por semana:	8
Ubicación en el plan de estudios:	Curso del quinto semestre de la carrera de Ingeniería en Computación
Requisitos:	IC5821 Requerimientos de Software
Correquisitos:	No los hay
El curso es requisito de:	IC6831 Aseguramiento de la Calidad del Software
Asistencia:	Obligatoria
Suficiencia:	No
Posibilidad de reconocimiento:	Si
Vigencia del programa:	II Semestre, 2012

2 Descripción general

En este curso se estudia el diseño de software, dentro del contexto del proceso de desarrollo de software. Se espera que al finalizar el curso el estudiante sea capaz de diseñar software mediante un proceso de diseño sistemático.

3 Objetivos

General

Valorar un conjunto de métodos, técnicas y herramientas para el diseño y la especificación de un producto de software.

Específicos

Al finalizar el curso el estudiante estará en capacidad de:

- Aplicar técnicas y herramientas orientadas a objetos para la modelación del diseño de software.
- Documentar la toma de decisiones durante la etapa de diseño del software.
- Comprender los diferentes niveles de abstracción en que deben expresarse las soluciones de problemas de diseño.
- Desarrollar destrezas para diseñar la arquitectura de software
- Analizar aspectos de las tecnologías actuales y de las tendencias tecnológicas que influyen en los diseños del software.

4 Contenidos

1) Introducción/vistazo al diseño y arquitectura de software. (0.5 semana)

a) De los requerimientos al diseño.

b) Pasos para diseñar software

c) Artefactos del diseño de software

d) El rol del arquitecto de software

2) El diseño de interfaces de usuario (3.5 semanas)

a) El concepto de utilizabilidad ("usability")

b) Técnicas para diseñar interfaces de usuario (diseño en papel, reutilización de conceptos y diseños conocidos, diseño con usuarios, etc.)

c) Diseño de la parte funcional de la interfaz, considerando diferentes tecnologías, eficiencia y eficacia

d) La importancia de la validación de datos

e) Tipos de componentes para el diseño de interfaces de usuario

f) Software para el diseño de interfaces de usuario

g) Guías de diseño gráfico de la interfaz, considerando diferentes tecnologías

h) Consideraciones para el diseño de reportes

3) Principios de diseño (1 semana)

a) Principio de divide y conquista

b) Principio de incrementar la cohesión

c) Principio de reducción de acoplamiento

d) Principio de mantenimiento del nivel de abstracción alto

e) Principio de incrementar la reusabilidad y reusar lo existente

f) Principio del diseño para la flexibilidad

g) Principio de anticipar la obsolescencia

h) Principio de diseñar para la portabilidad

i) Principio del Diseño para pruebas ("*testabilidad*")

j) Principio del diseño defensivo

4) El diseño de la arquitectura del software (5 semanas)

a) El entendimiento del problema

b) Identificación de elementos y sus relaciones

c) Descomposición del sistema

d) Diseño con componentes

e) Diseño con servicios

f) Diseño de la integración entre sistemas

g) El papel de los atributos de calidad (requerimientos no funcionales) en la especificación de la arquitectura

h) Especificación de la arquitectura de software (artefactos)

i) Estilos y patrones arquitectónicos

j) Tendencias tecnológicas en arquitectura de software

k) Arquitectura de aplicaciones empresariales

5) El diseño detallado del software (4 semanas)

a) Artefactos del diseño detallado

b) Diseño de los casos de uso

c) Diseño de los componentes y de los servicios

d) Diseño de clases

e) Aplicación de patrones de diseño en el diseño de clases

6) El proceso de revisión del diseño (1 semanas)

a) En relación con los requerimientos funcionales

b) En relación con los atributos de calidad del software (requerimientos no funcionales)

c) En relación con los requerimientos tecnológicos.

d) Revisión y Control del diseño durante el proceso de construcción de software.

7) Tendencias en el diseño de software (0.5 semana).

II parte: Aspectos operativos

5 Metodología de enseñanza y aprendizaje

Se emplearán técnicas de clases magistrales en su mayor parte presenciales, por parte del profesor, donde se desarrollarán los aspectos teóricos y prácticos más relevantes de los diferentes temas. Además, se combinarán con una alta participación por parte de los estudiantes durante el transcurso de las lecciones, por medio de llamadas orales, respuestas a casos en la pizarra y de trabajos en equipo. Algunas semanas al semestre se realizarán de forma remota, en las que puede haber sesión sincrónica o bien se destina para la elaboración de prácticas y trabajos en grupo de manera asincrónica con entregables definidos. Estas semanas están definidas previamente en el cronograma entregado con este programa de curso.

Los temas abordados en clase deben ser complementados por el estudiante a través de lecturas recomendadas por el profesor. Esto dará al estudiante un mayor criterio acerca de los contenidos presentados en el curso, para un mejor desempeño en el ámbito profesional.

A través del semestre, se desarrollará un proyecto en tres partes, donde se aplicarán las diferentes técnicas de diseño de sistemas de información automatizados, preferiblemente sobre una aplicación real. Los proyectos consideran la creación del diseño de software de una solución de computacional que responda a los requerimientos

establecidos al mismo tiempo que permite la extensión de nuevas funcionalidades. Adicionalmente el estudiante debe llevar a cabo la programación de los diseños propuestos. Debe asegurar que dicha implementación responda al diseño elaborado.

Actividades (quices, tareas, trabajos en clase)	15%
Prácticas Patrones (4)	20%
Investigación /Exposición	15%
Proyecto (1 en tres partes)	50%
TOTAL	**100%**

Ver detalle del cronograma en documento adjunto

7 Bibliografía

1. Primer: Agile Model-driven development with UML 2.0. Cambridge University Press.
2. Liskov, Barbara. 2001. Program Development in Java: Abstraction, Specification, and Object-oriented Design. Addison-Wesley.
3. Braude, Eric J. 2004. Software Design: from programming to architecture. John Wiley & Sons.
4. Lethbridge, Timothy C., Laganiere, Robert. 2nd Ed. 2004. Object-Oriented Software Engineering: Practical Software Development using UML and Java. McGrawHill.
5. Albin, Stephen. 2003. The Art of Software Architecture: Design Methods and Techniques. John Wiley & Sons.
6. Booch. Análisis y Diseño Orientado a Objetos Con Aplicaciones. Addison-Wesley/.

7. Budgen, David. Software Design, 2nd. Ed. Addison-Wesley, 2003
8. Fowler, M. UML Gota a Gota. Addison-Wesley, 1997.
9. Fowler, M. et. al. *Refactoring: Improving the Design of Existing Code*. Addison-Wesley, 2002.
10. Freeman, E. et. al. *Head First Design Patterns*. O´Reilly, 2002.
11. Gorton, Ian. Essential Software Architecture. Springer-Verlag, 2006.
12. Hamilton, K. Miles R. *Learning UML 2.0*. O´Reilly, 2006.
13. Jacobson; Booch; Rumbaugh. El Proceso Unificado de Desarrollo de Software. Addison-Wesley, 2000.
14. Jacobson; Booch; Rumbaugh. El Lenguaje Unificado de Modelado. Addison-Wesley, 1999.
15. IEEE Standard for Software Reviews, 1997.
16. Neil, Theresa. Mobile Design Pattern Gallery. O´Reilly 2nd. Ed. 2015.
17. Nielsen. Usability Engineering. Morgan Kaufmann. 1993
18. Nielsen. Designing Web Usability. New Riders. 2000
19. Sommerville. Ingeniería de Software. Addison-Wesley, 2000
20. Fairley, Richard,Ingeniería de Software, McGraw-Hill/Interamericana de México, S.A. de C.V., 1988.
21. Pressman,Ingeniería del Software, McGraw- Hill / interamericana de México, S.A. de C.V
22. James Rumbaugh, Objetct-Oriented Modeling and Design, Prentice Hall, 1991
23. Larman, Craig. UML y Patrones. 2da edición. Prentice Hall. 2003.
24. Bredemeyer, D. et al. The Role of the Architect in Software Development. Technical Report, HewlettPackard, Palo Alto, Calif. Bredemeyer. 2000.

8 Profesor

Ericka Solano Fernández tiene estudios de Ingeniería en Computación (ITCR) y Maestría en Administración Educativa (Universidad Latina de Costa Rica).

Ha laborado para la Escuela de Computación como docente desde el año 2001 y actualmente se desempeña como docente de la Escuela de Computación en el Centro Académico de San José.

Ha impartido cursos como Fundamentos de Organización de Computadoras, Introducción a la Programación, Taller de Programación, Estructuras de Datos, Programación Orientada a Objetos, Lenguajes de Programación, Principios de Sistemas Operativos, Requerimientos de Software, Diseño de Software, Aseguramiento de la Calidad de Software, Análisis y Diseño de Algoritmos, Introducción al desarrollo de aplicaciones WEB, Elementos de Computación para la Escuela de Computación.

Ha impartido cursos de programación y metodologías de desarrollo de software en otras instituciones educativas desde el año 2006 y ha sido instructora en cursos de capacitación de actualización profesional en el sector público y privado en temas de modelaje de aplicaciones, programación en Java Básico, Intermedio y Avanzado, talleres de requerimientos con UML 2.0, así como aplicaciones WEB.

Correo electrónico de contacto oficial: **ersolano@itcr.ac.cr**
Contacto vía Skype: **ericka.solano**
Medio oficial electrónico: TEC-Digital (http://tecdigital.tec.ac.cr)

Grupo de comunicación en Whatsapp del curso (ver TEC Digital):
CTLSJ: https://chat.whatsapp.com/Lhd5HuukFacLGLPrSABkAa
Cartago: https://chat.whatsapp.com/ELmpYHzvzes0AlEGSyJz0b

Consulta presencial:

CTLSJ : Martes y Jueves 3:00-4:00p.m. en Casa Verde con con cita previa o según acuerdo entre profesora y estudiante.

Se atenderán consultas por mensajería instantánea (whatsapp) dentro de las posibilidades de atención de la profesora.